Aggregate functions perform a calculation on a set of values and return a single value. Except for COUNT, aggregate functions ignore null values. Aggregate functions are frequently used with the GROUP BY clause of the SELECT statement.

All aggregate functions are deterministic. This means aggregate functions return the same value any time that they are called by using a specific set of input values. For more information about function determinism, see Deterministic and Nondeterministic Functions. The OVER clause may follow all aggregate functions except GROUPING and GROUPING\_ID.

Aggregate functions can be used as expressions only in the following:

The select list of a SELECT statement (either a subquery or an outer query).

Transact-SQL provides the following aggregate functions:

AVG

MIN

CHECKSUM\_AGG

SUM

COUNT STDEV

COUNT\_BIG

STDEVP

GROUPING

VAR

GROUPING\_ID VARP

MAX

-------------------------------------------------------------------------

**In SQL server, there are different types of JOINS.**  
1. CROSS JOIN  
2. INNER JOIN   
3. OUTER JOIN   
  
**Outer Joins are again divided into 3 types**  
1. Left Join or Left Outer Join  
2. Right Join or Right Outer Join  
3. Full Join or Full Outer Join

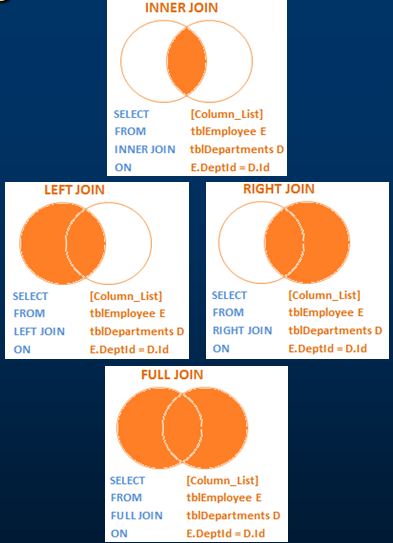
**General Formula for Joins**  
SELECT      ColumnList  
FROM           LeftTableName  
JOIN\_TYPE  RightTableName  
ON                JoinCondition

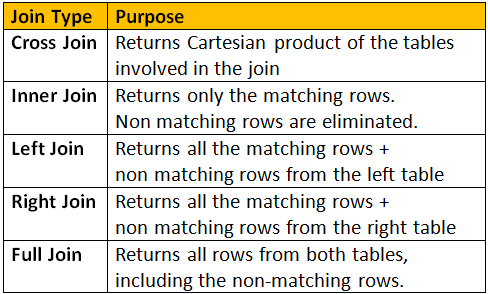
INNER JOIN, returns only the matching rows between both the tables. Non matching rows are eliminated.

**Note:** You can use, LEFT JOIN or LEFT OUTER JOIN. OUTER keyowrd is optional  
  
**LEFT JOIN**, returns all the matching rows + non matching rows from the left table. In reality, INNER JOIN and LEFT JOIN are extensively used.

**Note:** You can use, RIGHT JOIN or RIGHT OUTER JOIN. OUTER keyowrd is optional  
  
**RIGHT JOIN**, returns all the matching rows + non matching rows from the right table.

**Note:** You can use, FULLJOIN or FULL OUTER JOIN. OUTER keyowrd is optional  
  
**FULL JOIN**, returns all rows from both the left and right tables, including the non matching rows.

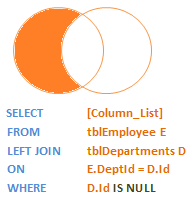




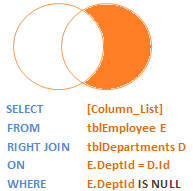
**SQL Cartesian Product** Tips. The **Cartesian product**, also referred to as a cross-join, returns all the rows in all the tables listed in the query. Each row in the first table is paired with all the rows in the second table. This happens when there is no relationship defined between the two tables.

Advanced Joins

1. Advanced or intelligent joins in SQL Server  
2. Retrieve only the non matching rows from the left table  
3. Retrieve only the non matching rows from the right table  
4. Retrieve only the non matching rows from both the left and right table

SELECT       Name, Gender, Salary, DepartmentName  
FROM           tblEmployee E  
LEFT JOIN   tblDepartment D  
ON                 E.DepartmentId = D.Id  
WHERE        D.Id IS NULL  
  


**How to retrieve only the non matching rows from the right table**  
http://1.bp.blogspot.com/-ycu3SCB-mn0/UDDj12WVhYI/AAAAAAAAAP4/WoJn5XGOlJc/s1600/Only+Right+Table+Rows.png   
  
**Query:**  
SELECT         Name, Gender, Salary, DepartmentName  
FROM             tblEmployee E  
RIGHT JOIN    tblDepartment D  
ON                   E.DepartmentId = D.Id  
WHERE          E.DepartmentId IS NULL





Your just adding a where clause to make the join ‘Intelligent’

**Never Use = to compare to find a null column value**

**Always use Is**

**Where accounted is null**

**Self Join Query:**   
A MANAGER is also an EMPLOYEE. Both the, EMPLOYEE and MANAGER rows, are present in the same table. Here we are joining tblEmployee with itself using different alias names, E for Employee and M for Manager. We are using LEFT JOIN, to get the rows with ManagerId NULL. You can see in the output TODD's record is also retrieved, but the MANAGER is NULL. If you replace LEFT JOIN with INNER JOIN, you will not get TODD's record.  
Select E.Name as Employee, M.Name as Manager  
from tblEmployee E  
Left Join tblEmployee M  
On E.ManagerId = M.EmployeeId  
  
  
In short, joining a table with itself is called as **SELF JOIN**. SELF JOIN is not a different type of JOIN. It can be classified under any type of JOIN - INNER, OUTER or CROSS Joins. The above query is, LEFT OUTER SELF Join.

**Inner Self Join tblEmployee table:**  
Select E.Name as Employee, M.Name as Manager  
from tblEmployee E  
Inner Join tblEmployee M  
On E.ManagerId = M.EmployeeId  
  
**Cross Self Join tblEmployee table:**  
Select E.Name as Employee, M.Name as Manager  
from tblEmployee  
Cross Join tblEmployee